UPM ETSIInf. Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software.

Examen Parcial de Programación II – Ejercicio Práctico

18 de Mayo de 2015

Duración: La duración total del ejercicio será de **1 hora y 15 minutos**.
Calificaciones: Las notas se publicarán el día **25 de Mayo**
Revisión: La revisión tendrá lugar los días **26, 27 y 28 de Mayo** de 11:00h a 13:00h en el despacho D-2315**.**
Hoja de respuestas: El código solicitado debe escribirse en los espacios señalados en la **hoja de respuestas**. Pueden usarse hojas en blanco adicionales como borrador. Sólo hay que entregar la hoja de respuestas.

Ejercicio Práctico

Se trata de desarrollar parte del código de un sistema simplificado y rudimentario de manejo de cuentas bancarias. Se dispone de **código de apoyo** ya desarrollado de:

- Clase **`Cuenta`** sin restricciones, permite siempre ingresar y retirar, incluso con descubierto.
- Esqueleto de una clase **`Banco`** que debe gestionar una colección de cuentas en un vector. Las cuentas pueden ser de la clase `Cuenta` o derivadas, estar en cualquier posición, y haber huecos a `null`.
- Excepción `ExcepcionOperacionNoValida`.

El código concreto a desarrollar es el siguiente:

1. **Clase `CuentaDeposito`**, es hija de la clase Cuenta con un atributo booleano adicional *vencido* (que indica si el depósito ha vencido o no). **En una CuentaDepósito no se puede nunca ingresar dinero**. Al crear una `CuentaDeposito` el atributo *vencido* se inicializará a falso.

   Se implementará un nuevo método `marcarVencido()` que cambiará el valor de *vencido* a cierto. A este método se le llamará cuando haya pasado la fecha de vencimiento de la CuentaDeposito. A partir que haya vencido el plazo del depósito se podrá retirar dinero de la cuenta (siempre que haya suficiente saldo).

   Por tanto hay que declarar el atributo adicional *vencido*, implementar el método `marcarVencido()`, y redefinir los métodos `ingresar()` y `retirar()` para que lancen la excepción `ExcepcionOperacionNoValida` si se intenta realizar alguna operación de las que no están permitidas según la descripción de `CuentaDeposito`.

2. **Método `saldoTotal(titular)` en la clase Banco**, que recorra la colección de cuentas y devuelva la suma de saldos de las cuentas del titular indicado como argumento. Obviamente, el resultado será cero si no hay ninguna cuenta de ese titular.

   Las cuentas pueden estar en cualquier posición, y haber "huecos" entre ellas ocupados con `null`.

3. **Método `ingresar(importe, titular)` en la clase Banco**, que recorra la colección de cuentas e ingrese el importe en la primera cuenta de ese titular que admita ingresos. Si no hay ninguna cuenta de ese titular o solo hay cuentas que no admiten ingresos, se lanzará la excepción `ExcepcionOperacionNoValida`.

   Sugerencias: Con cada cuenta del titular, comprobar que no es CuentaDeposito antes de intentar el ingreso (o bien intentar el ingreso y si se lanza la excepción, capturarla sin hacer nada y seguir buscando la siguiente cuenta). En cualquier caso si el ingreso tiene éxito el recorrido termina inmediatamente.

Código de apoyo

```
public class ExcepcionOperacionNoValida extends Exception {....}
```

```java
// Entidad bancaria (colección de cuentas)
public class Banco {
    private Cuenta[] cuentas;
        ....
    // Consultar saldo total
    public int saldoTotal(String titular) {
        // TODO
        ....
    }
    // Realizar ingreso
    public void ingresar(int importe, String titular)
            throws ExcepcionOperacionNoValida {
        ....
    }
}
```

```java
// Cuenta bancaria simple (cuenta corriente)
public class Cuenta {
    private String titular;
    private int saldo;

    // Constructor
    public Cuenta(String titular, int saldoInicial)
            throws ExcepcionOperacionNoValida {
        if (titular == null || saldoInicial <= 0)
            throw new ExcepcionOperacionNoValida();
        this.titular = titular;
        saldo = saldoInicial;
    }

    // Getters
    public String getTitular() {
        return titular;
    }
    public int getSaldo() {
        return saldo;
    }

    // Ingresar un importe
    public void ingresar(int importe) throws ExcepcionOperacionNoValida {
        if (importe <= 0)
            throw new ExcepcionOperacionNoValida();
        saldo = saldo + importe;
    }

    // Retirar un importe
    public void retirar(int importe) throws ExcepcionOperacionNoValida {
        if (importe <= 0)
            throw new ExcepcionOperacionNoValida();
        saldo = saldo - importe;
    }
}
```

UPM ETSIInf. Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software.

Examen Parcial de Programación II – Ejercicio Práctico

18 de Mayo de 2015 – HOJA DE RESPUESTA

Apellidos _____

Nombre _____

DNI/NIE/Pasaporte _____ **Nº Matrícula** _____

Clase `CuentaDeposito`: Cabecera de la clase, atributo adicional, constructor, métodos `ingresar()`, `retirar()` y `marcarVencido()`.

Método `saldoTotal(titular)` de la clase `Banco`.

Método `ingresar(importe, titular)` de la clase `Banco`.